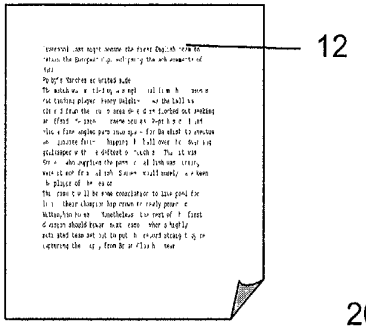

Fig. 1A

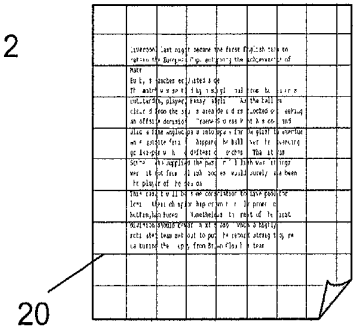

Fig. 1B

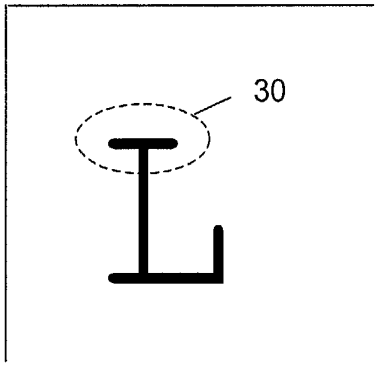

Fig. 1C

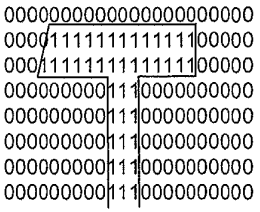

Fig. 1D

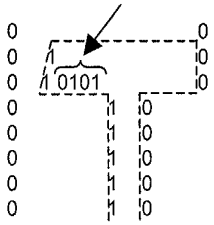

Fig. 1E

New U.S. Patent Application Filed August 29, 2001
HP No. 30006581
METHOD AND APPARATUS FOR EMBODYING DOCUMENTS
John W. Ryan (202) 663-6446
WILMER, CUTLER & PICKERING
Sheet 1 of 9

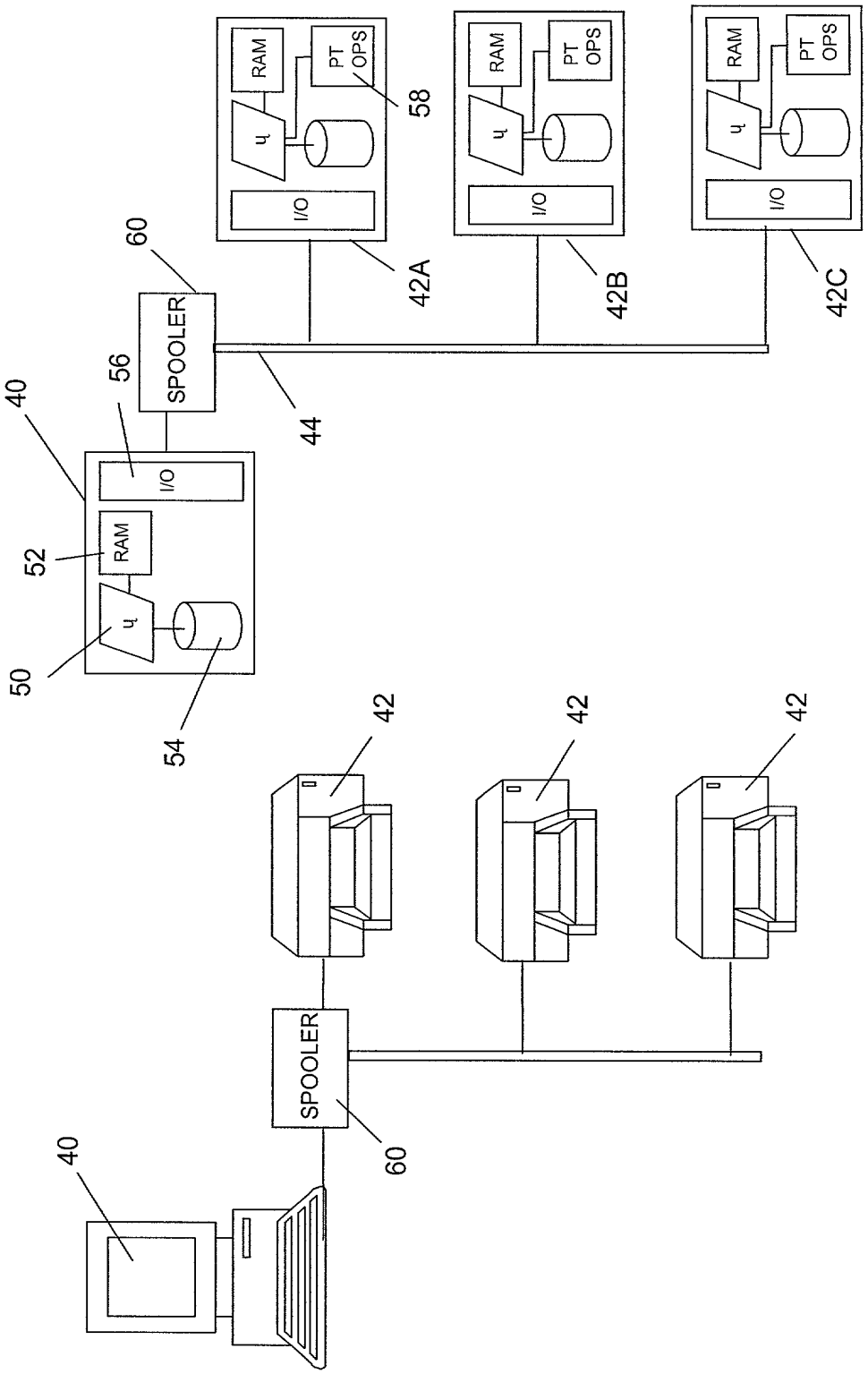
RAM
PT OPS
I/O
42A
42B
42C
58
60
SPOOLER
56
44
40
52
50
54
42
42
42
SPOOLER
60
40

Fig.2B

Fig. 2A

```
/*
  Skeleton rip_nprint, send and status functions, written in a C like
languages,
  all of which rely on POSIX compliant i/o substructure
*/

/*
 Skeleton rip_nprint function

 on completion, returns a pointer to data structure that represents
 the transformed data
*/

/*
  success of ripping operation
*/
enum success {
  done,      /* rip complete */
  fail,      /* rip failed */
  incomplete   /* rip incomplete */
};

/*
  result of ripping operation
*/
struct rip_pcl {
  enum sucess result; /* result of operation */
  rip  *data
};

/*
 rip_nprint
*/
enum success rip_nprint(pcl *raw_data, char *rip_name, char *my_name){
  rip_pcl *completed; /* data once operated on */
  network_id source, destination;
  socket_interface dest_printer;

  setup(completed); /* allocate space for completed data structure */
  source = return_net_id(my_name);
  destination = return_net_id(rip_name);
  dest_printer = open_socket(destination, source);
  if (dest_printer != NULL) {
   if ((transmit(dest_printer, data) != NULL) {
    return (success);
   } else
   {
     return (fail);
   }
  } else
  {
   return(fail);
  }
}
```

Fig. 3

```
/*
 skeleton send function

 takes name of destination to which data must be transmitted, calls i/o
function
 transmit, and returns a success flag
*/

enum success send_rip(rip *data, char * dest_name, char * my_name) {
  network_id source, destination;
  socket_interface dest_device;

  source = return_net_id(my_name);
  destination = return_net_id(dest_name);
  dest_device = open_socket(destination, source);
  if (dest_printer != NULL) {
   if ((transmit(dest_printer, data) != NULL) {
    return (success);
   } else
   {
     return (fail);
   }
  } else
  {
   return(fail);
  }
}
```

Fig. 4

```
/*
 job data structure
*/

struct job {
 long_int priority;
 long_int time_to_complete;
 long_int memory_to_complete;
};

/*
  status data structure (minimal), leaving out temp, paper handling,
  error and wear statuis
*/


struct status {
  struct job job_list[max_jobs];
};

/*
  skeleton status function
*/

struct status *get_status(char *rip_name, char *my_name) {
  network_id source, destination;
  socket_interface dest_device;
  struct job *this_job;

  job = allocate_jop_status();
  source = return_net_id(my_name);
  destination = return_net_id(dest_name);
  dest_device = open_socket(destination, source);

  job = call_remote_function(dest_device, status);

  return(job);
}
```

Fig. 5

```
/*
 Skeleton suspend_n_rip function

 suspends the current RIP job, allocates space for the job, saves it,
 rips new job and then recovers old job
*/


pcl *current_data;   /* pointer to storage space for suspended job */

void suspend_n_rip(pcl *raw_data){
  wait_on_break(current_data); /* waits until  job can be broken
(page/batch boundary) */
  suspend(current_data) /* purges machinbe of data and data specific
settings */
  print(raw_data);      /* rips and prints the new job */
  restore(current_data);/* restore current job */
}
```

Fig. 6

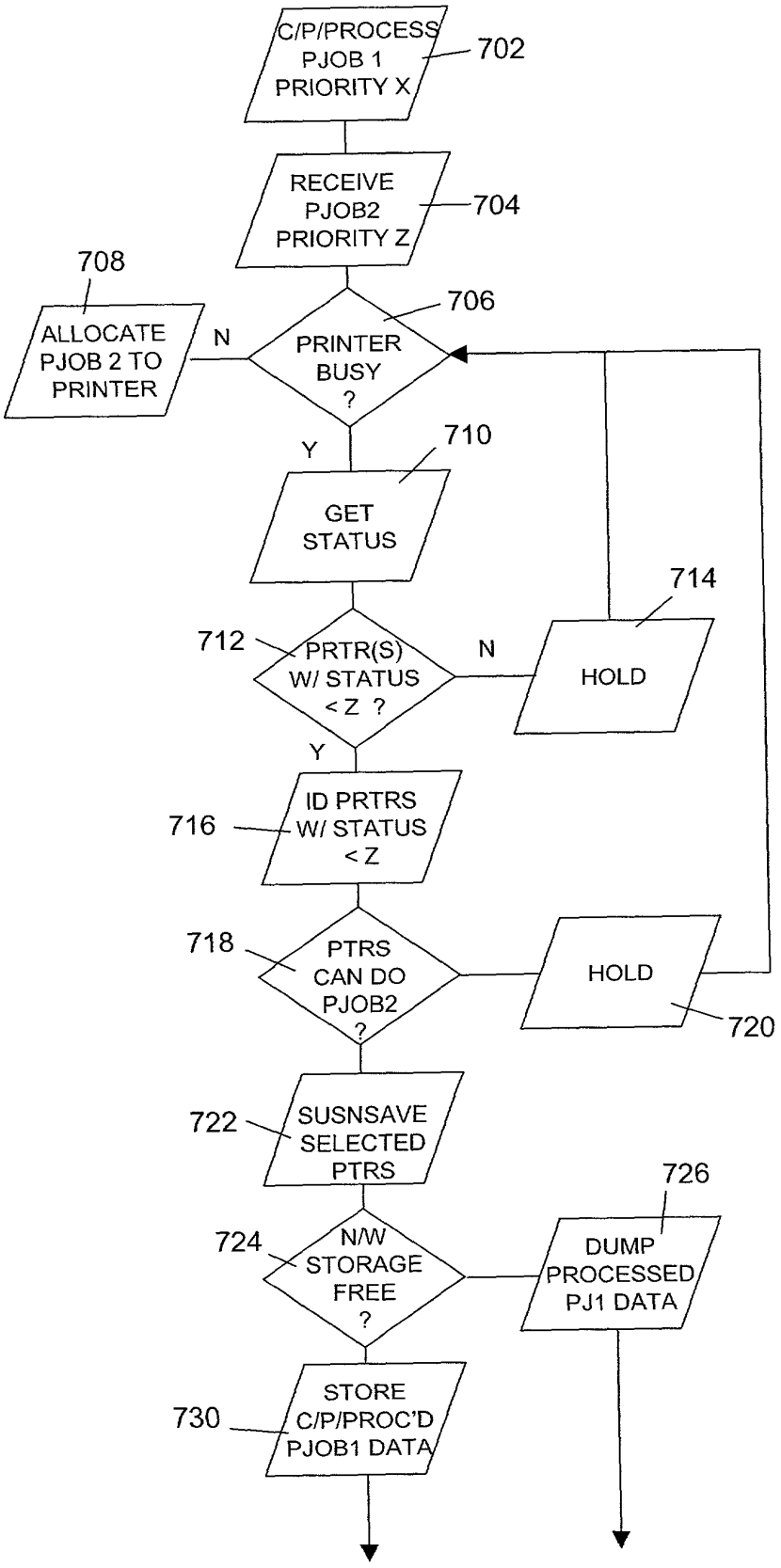
C/P/PROCESS PJOB 1 PRIORITY X
702
RECEIVE PJOB2 PRIORITY Z
704
708
ALLOCATE PJOB 2 TO PRINTER
N
706
PRINTER BUSY ?
Y
710
GET STATUS
714
712
PRTR(S) W/ STATUS < Z ?
N
HOLD
Y
ID PRTRS W/ STATUS < Z
716
718
PTRS CAN DO PJOB2 ?
HOLD
720
SUSNSAVE SELECTED PTRS
722
726
724
N/W STORAGE FREE ?
DUMP PROCESSED PJ1 DATA
STORE C/P/PROC'D PJOB1 DATA
730

Fig. 7

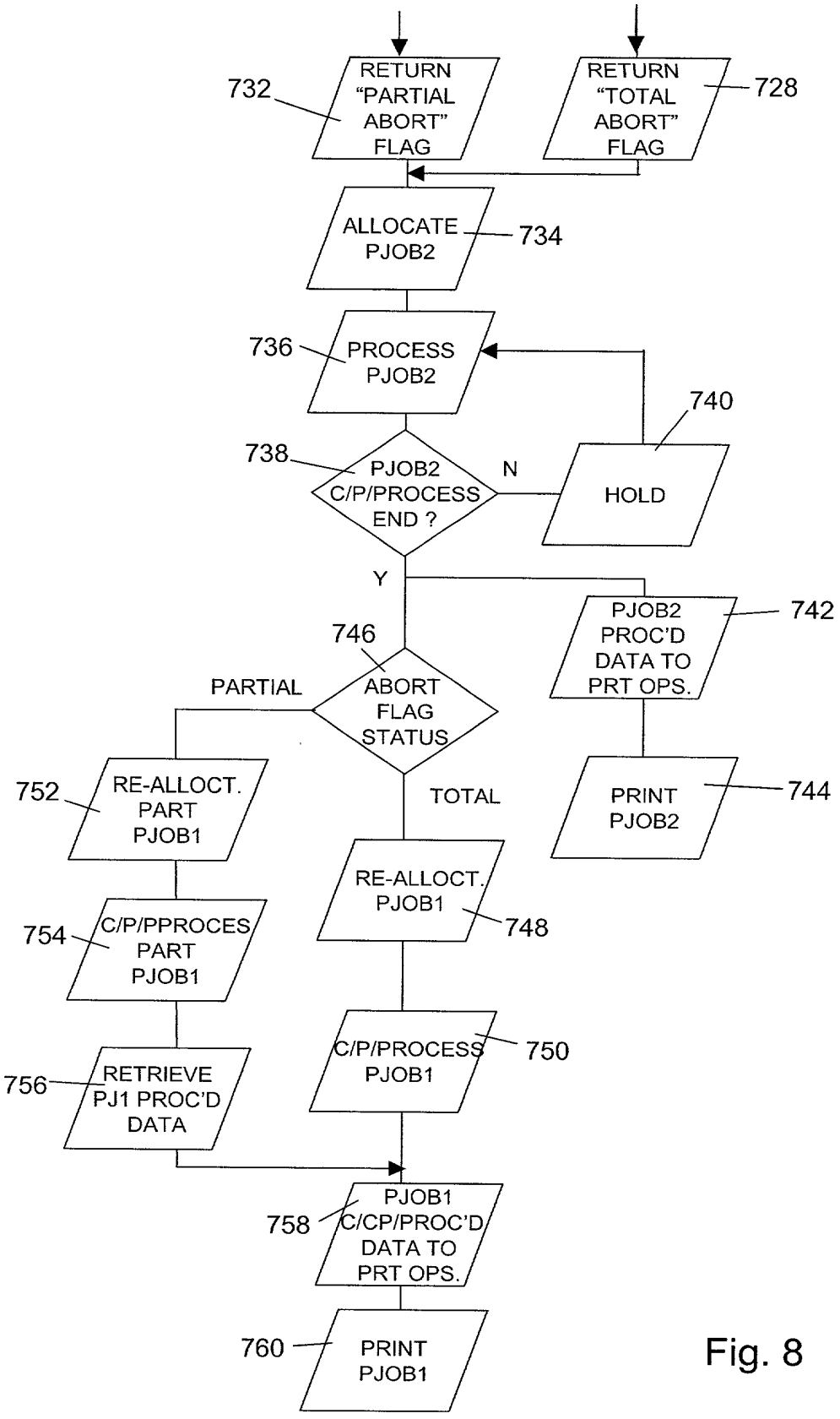
RETURN "PARTIAL ABORT" FLAG
732
RETURN "TOTAL ABORT" FLAG
728
ALLOCATE PJOB2
734
PROCESS PJOB2
736
740
PJOB2 C/P/PROCESS END ?
738
N
HOLD
Y
PJOB2 PROC'D DATA TO PRT OPS.
742
746
ABORT FLAG STATUS
PARTIAL
TOTAL
PRINT PJOB2
744
RE-ALLOCT. PART PJOB1
752
RE-ALLOCT. PJOB1
748
C/P/PPROCES PART PJOB1
754
C/P/PROCESS PJOB1
750
RETRIEVE PJ1 PROC'D DATA
756
PJOB1 C/CP/PROC'D DATA TO PRT OPS.
758
PRINT PJOB1
760

Fig. 8

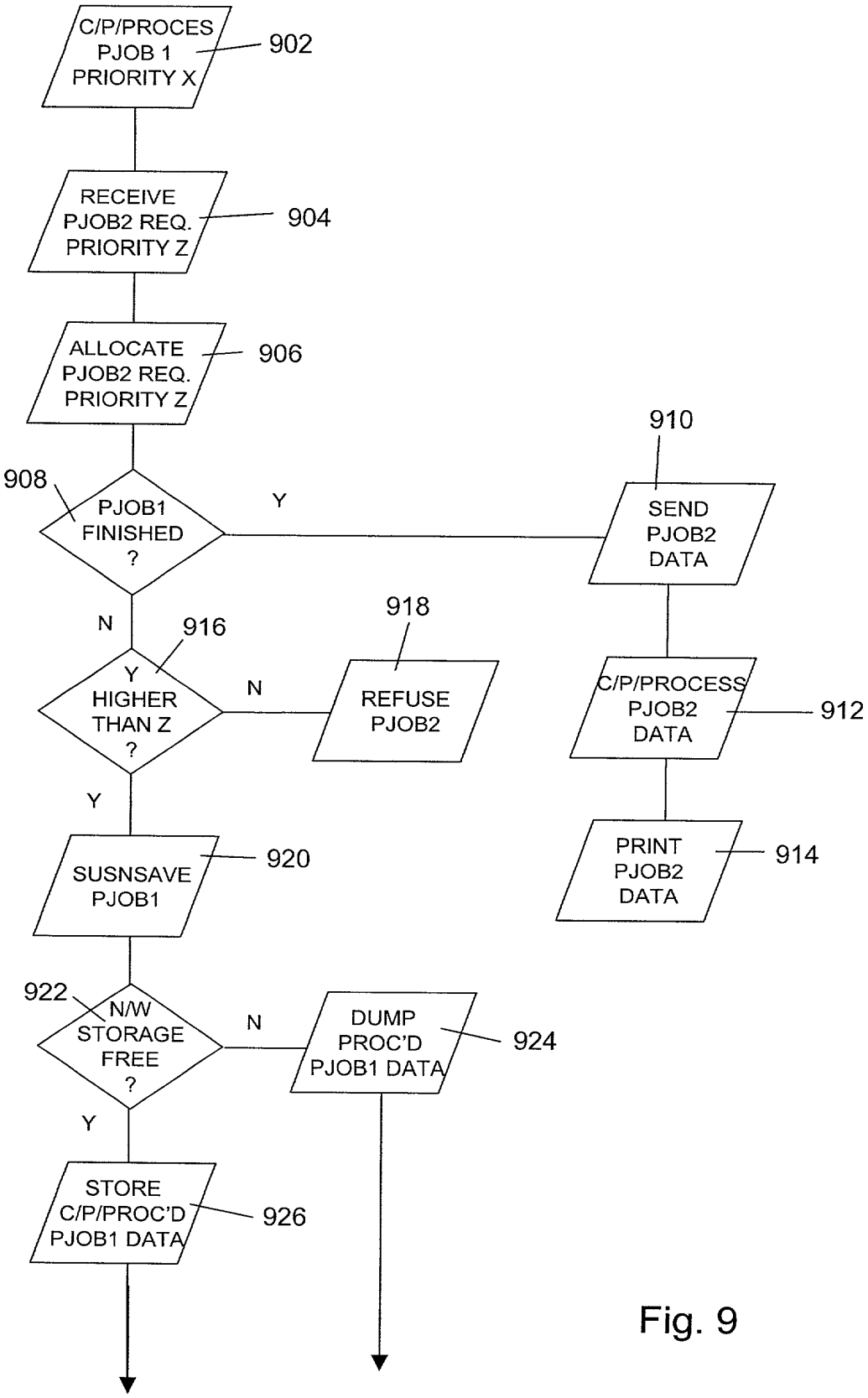
C/P/PROCES PJOB 1 PRIORITY X
902
RECEIVE PJOB2 REQ. PRIORITY Z
904
ALLOCATE PJOB2 REQ. PRIORITY Z
906
908
PJOB1 FINISHED ?
Y
910
SEND PJOB2 DATA
N
916
Y
HIGHER THAN Z ?
N
918
REFUSE PJOB2
C/P/PROCESS PJOB2 DATA
912
Y
SUSNSAVE PJOB1
920
PRINT PJOB2 DATA
914
922
N/W STORAGE FREE ?
N
DUMP PROC'D PJOB1 DATA
924
Y
STORE C/P/PROC'D PJOB1 DATA
926

Fig. 9